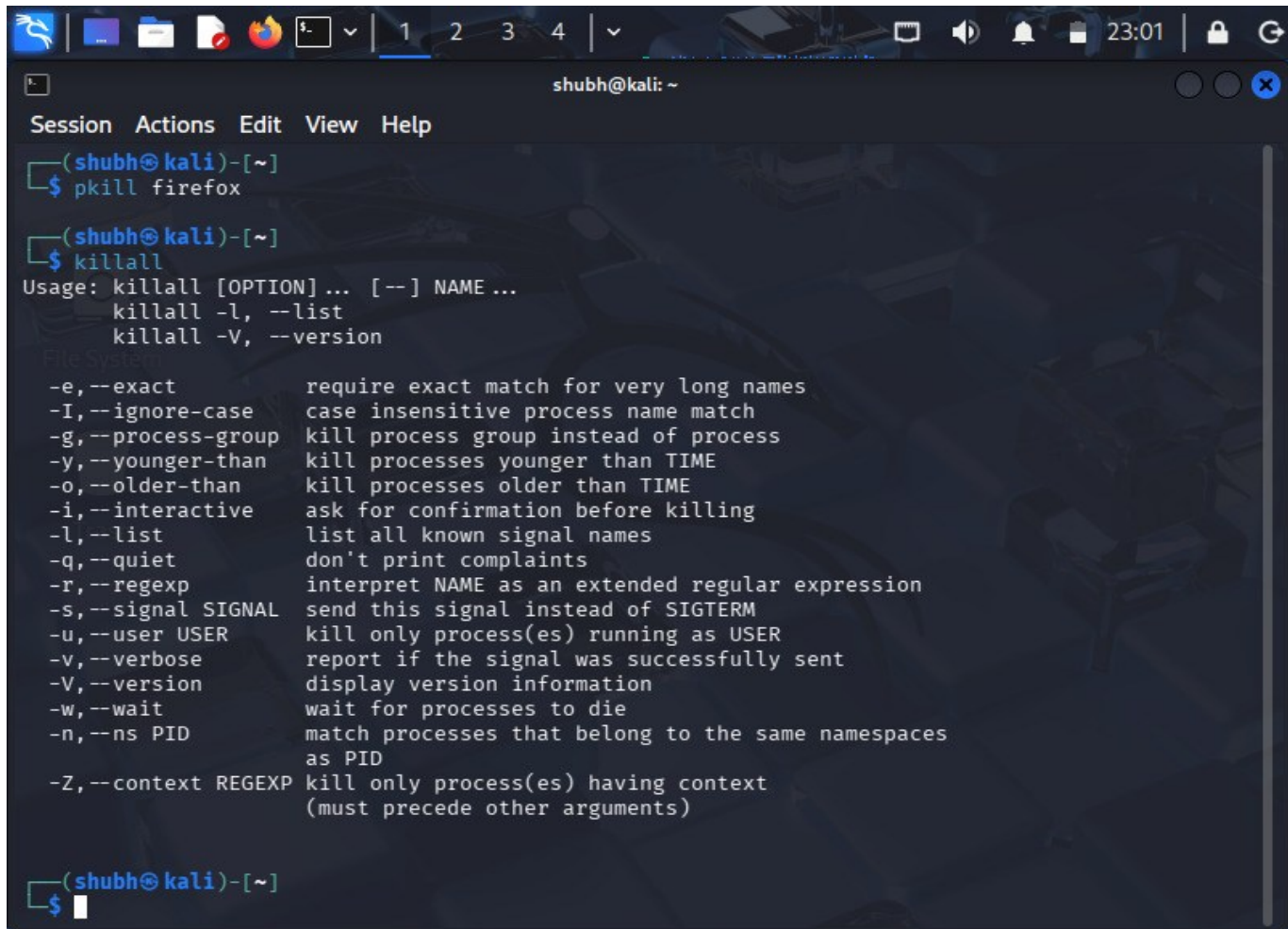
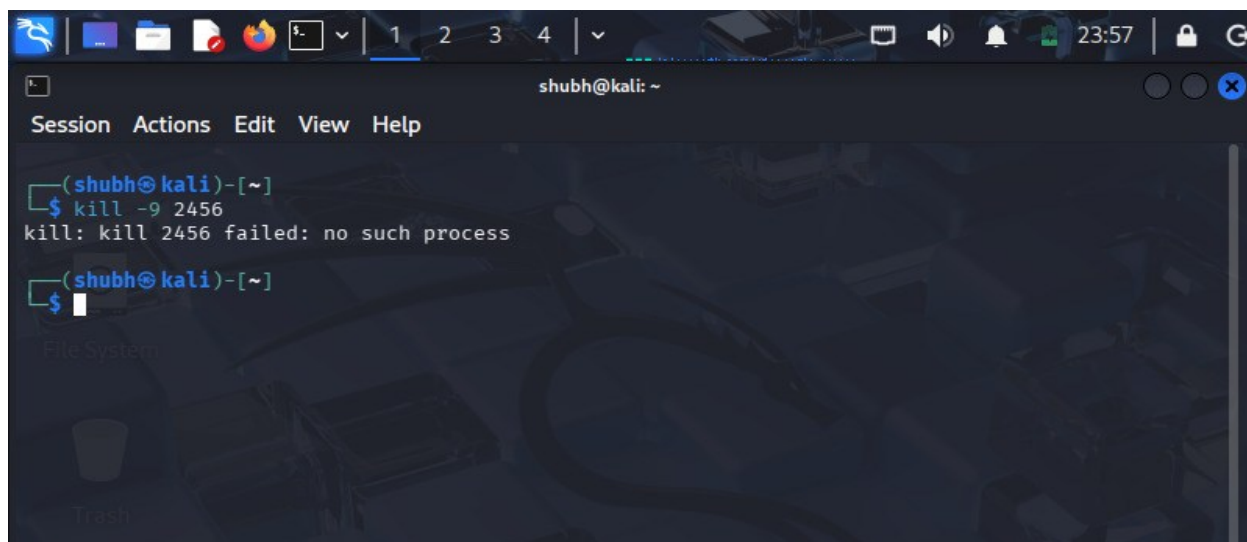


1. Kill Processes by name
2. Kill a process based on the process name .
3. Kill a single process at a time with the given process ID



A terminal window titled 'shubh@kali: ~' with a menu bar (Session, Actions, Edit, View, Help). The user enters the command `pkill firefox`. Then, they enter `killall`, which triggers a usage message and a list of options. The options include `-e, --exact` (require exact match for very long names), `-I, --ignore-case` (case insensitive process name match), `-g, --process-group` (kill process group instead of process), `-y, --younger-than` (kill processes younger than TIME), `-o, --older-than` (kill processes older than TIME), `-i, --interactive` (ask for confirmation before killing), `-l, --list` (list all known signal names), `-q, --quiet` (don't print complaints), `-r, --regex` (interpret NAME as an extended regular expression), `-s, --signal SIGNAL` (send this signal instead of SIGTERM), `-u, --user USER` (kill only process(es) running as USER), `-v, --verbose` (report if the signal was successfully sent), `-V, --version` (display version information), `-w, --wait` (wait for processes to die), `-n, --ns PID` (match processes that belong to the same namespaces as PID), and `-Z, --context REGEXP` (kill only process(es) having context (must precede other arguments)).

```
(shubh@kali)-[~]  
$ pkill firefox  
  
(shubh@kali)-[~]  
$ killall  
Usage: killall [OPTION]... [--] NAME ...  
    killall -l, --list  
    killall -V, --version  
  
File System  
-e,--exact          require exact match for very long names  
-I,--ignore-case    case insensitive process name match  
-g,--process-group  kill process group instead of process  
-y,--younger-than   kill processes younger than TIME  
-o,--older-than     kill processes older than TIME  
-i,--interactive    ask for confirmation before killing  
-l,--list           list all known signal names  
-q,--quiet          don't print complaints  
-r,--regex          interpret NAME as an extended regular expression  
-s,--signal SIGNAL  send this signal instead of SIGTERM  
-u,--user USER      kill only process(es) running as USER  
-v,--verbose        report if the signal was successfully sent  
-V,--version        display version information  
-w,--wait           wait for processes to die  
-n,--ns PID         match processes that belong to the same namespaces  
                    as PID  
-Z,--context REGEXP kill only process(es) having context  
                    (must precede other arguments)  
  
(shubh@kali)-[~]  
$
```

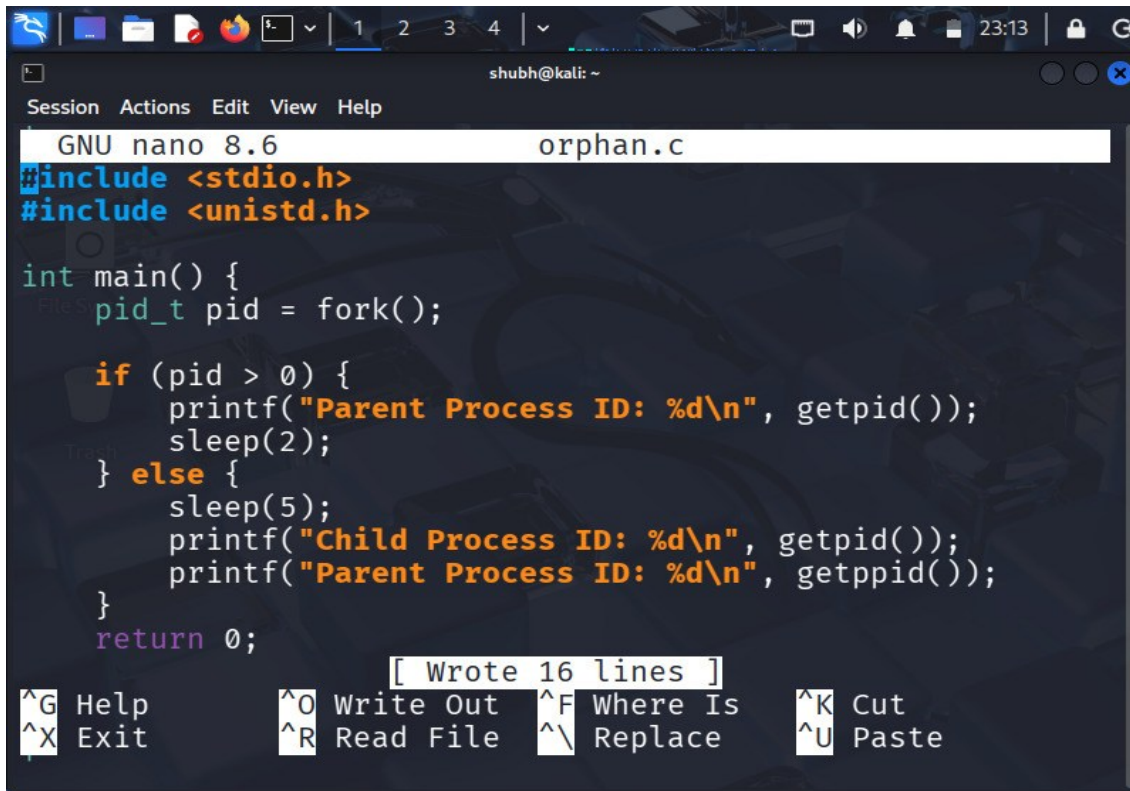


A terminal window titled 'shubh@kali: ~' with a menu bar (Session, Actions, Edit, View, Help). The user enters the command `kill -9 2456`. The terminal displays the error message: `kill: kill 2456 failed: no such process`. The user then enters a new command, which is partially visible as `$`.

```
(shubh@kali)-[~]  
$ kill -9 2456  
kill: kill 2456 failed: no such process  
  
(shubh@kali)-[~]  
$
```

2. Write a program for process creation using C

1. Orphan Process 2. Zombie process



The screenshot shows a terminal window with the nano text editor open, editing a file named `orphan.c`. The code defines a `main` function that uses `fork()` to create a child process. The parent process prints its ID and sleeps for 2 seconds. The child process prints its ID and the parent's ID using `getppid()` and sleeps for 5 seconds. The status bar at the bottom indicates that 16 lines have been written.

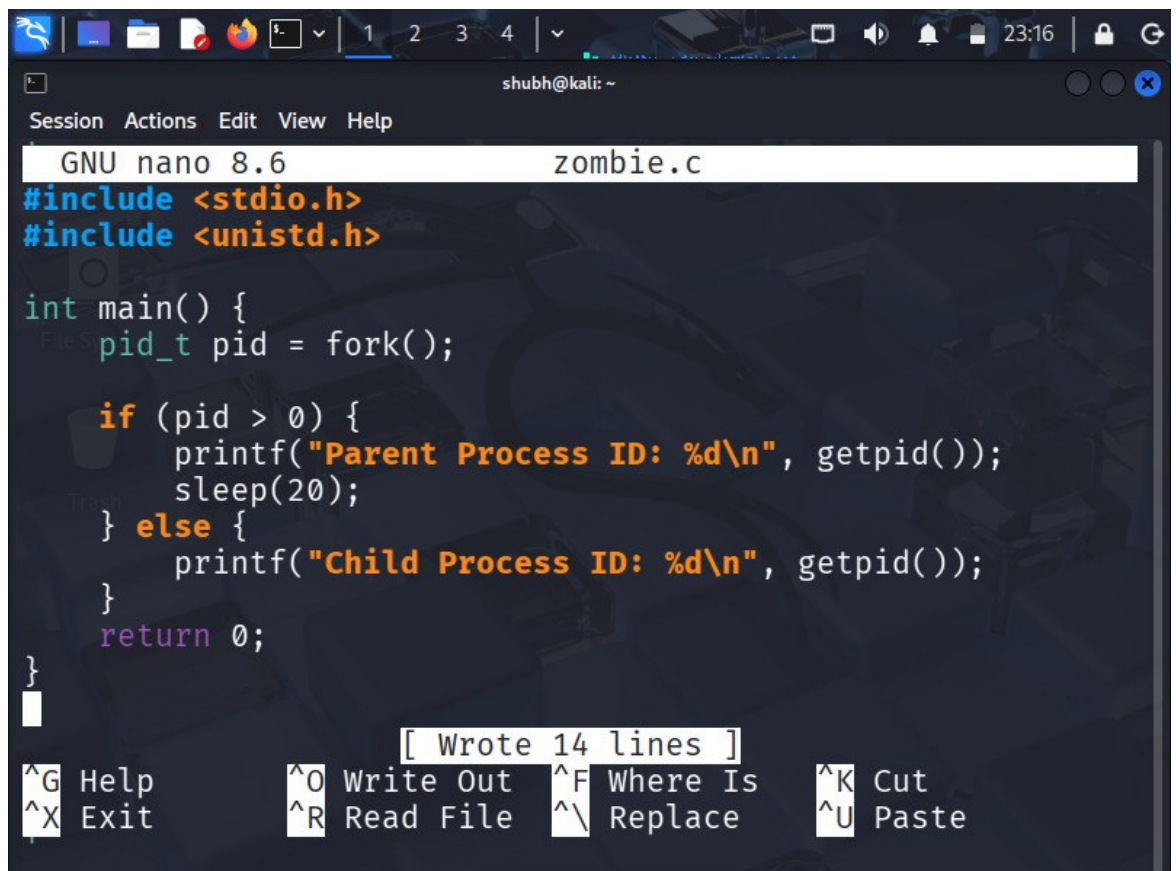
```
GNU nano 8.6 orphan.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        printf("Parent Process ID: %d\n", getpid());
        sleep(2);
    } else {
        sleep(5);
        printf("Child Process ID: %d\n", getpid());
        printf("Parent Process ID: %d\n", getppid());
    }
    return 0;
}
```

[Wrote 16 lines]

Help Write Out Where Is Cut
Exit Read File Replace Paste



The screenshot shows a terminal window with the nano text editor open, editing a file named `zombie.c`. The code defines a `main` function that uses `fork()` to create a child process. The parent process prints its ID and sleeps for 20 seconds. The child process prints its ID. The status bar at the bottom indicates that 14 lines have been written.

```
GNU nano 8.6 zombie.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        printf("Parent Process ID: %d\n", getpid());
        sleep(20);
    } else {
        printf("Child Process ID: %d\n", getpid());
    }
    return 0;
}
```

[Wrote 14 lines]

Help Write Out Where Is Cut
Exit Read File Replace Paste

3. Create the process using fork()system cell

1.Child Process creation 2.Parent Process creation

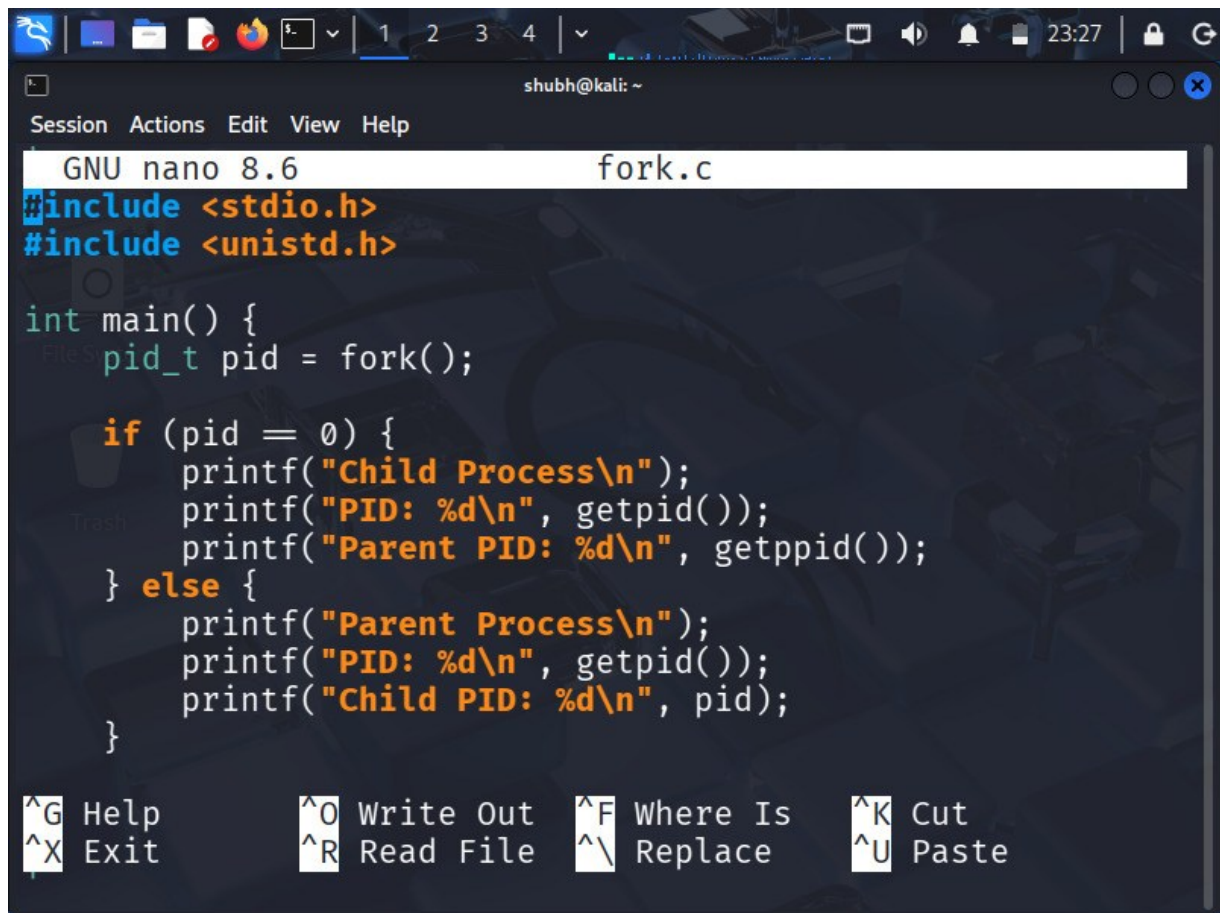
```
shubh@kali: ~  
Session Actions Edit View Help  
└─$ gcc orphan.c -o orphan  
cc1: fatal error: orphan.c: No such file or directory  
compilation terminated.  
  
└─(shubh@kali)-[~]  
└─$ ls  
Desktop Documents Downloads Music Pictures process.c  
  
└─(shubh@kali)-[~]  
└─$ gcc process.c -o orphan  
  
└─(shubh@kali)-[~]  
└─$ nano orphan.c  
  
└─(shubh@kali)-[~]  
└─$ gcc process.c -o orphan  
  
└─(shubh@kali)-[~]  
└─$
```

```
shubh@kali: ~  
Session Actions Edit View Help  
└─$ gcc process.c -o orphan  
  
└─(shubh@kali)-[~]  
└─$ nano orphan.c  
  
└─(shubh@kali)-[~]  
└─$ gcc process.c -o orphan  
  
└─(shubh@kali)-[~]  
└─$ nano zombie.c  
  
└─(shubh@kali)-[~]  
└─$ gcc zombie.c -o zombie  
  
└─(shubh@kali)-[~]  
└─$ ./zombie  
Parent Process ID: 21580  
Child Process ID: 21581  
└─
```



```
shubh@kali: ~  
Session Actions Edit View Help  
(shubh@kali)-[~]  
$ nano zombie.c  
  
(shubh@kali)-[~]  
$ gcc zombie.c -o zombie  
File System  
(shubh@kali)-[~]  
$ ./zombie  
Parent Process ID: 21580  
Child Process ID: 21581  
  
(shubh@kali)-[~]  
$ ./orphan  
Parent Process ID: 22125  
  
(shubh@kali)-[~]  
$ Child Process ID: 22126  
Parent Process ID: 1  
█
```

```
shubh@kali: ~  
Session Actions Edit View Help  
nano fork.c  
  
(shubh@kali)-[~]  
$ nano fork.c  
  
(shubh@kali)-[~]  
$ gcc fork.c -o fork  
  
(shubh@kali)-[~]  
$ ./fork  
Parent Process  
PID: 26532  
Child PID: 26533  
  
Child Process  
PID: 26533  
Parent PID: 1  
(shubh@kali)-[~]  
$ █
```



The image shows a terminal window with a dark background. At the top, there is a taskbar with various icons and a system clock showing 23:27. The terminal window title is "shubh@kali: ~". The menu bar includes "Session", "Actions", "Edit", "View", and "Help". The editor is GNU nano 8.6, editing a file named "fork.c". The code is as follows:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child Process\n");
        printf("PID: %d\n", getpid());
        printf("Parent PID: %d\n", getppid());
    } else {
        printf("Parent Process\n");
        printf("PID: %d\n", getpid());
        printf("Child PID: %d\n", pid);
    }
}
```

At the bottom of the terminal, there is a help section with the following shortcuts:

^G Help	^O Write Out	^F Where Is	^K Cut
^X Exit	^R Read File	^\\ Replace	^U Paste

